Jointly Optimal Routing and Caching with Bounded Link Capacities

Yuanyuan Li,[1] Yuchao Zhang,[3,2] Stratis Ioannidis,[1] and Jon Crowcroft[2]
[1]Northeastern University, [2]University of Cambridge, [3]Beijing University of Posts and Telecommunications
Email: yuanyuanli@ece.neu.edu, yczhang@bupt.edu.cn, ioannidis@ece.neu.edu, jon.crowcroft@cl.cam.ac.uk

***Abstract*—We study a cache network in which intermediate nodes equipped with caches can serve requests. We model the problem of jointly optimizing caching and routing decisions with link capacity constraints over an arbitrary network topology. This problem can be formulated as a continuous diminishing-returns (DR) submodular maximization problem under multiple continuous DR-supermodular constraints, and is NP-hard. We propose a poly-time alternating primal-dual heuristic algorithm, in which primal steps produce solutions within $1-\frac{1}{e}$ approximation factor from the optimal. Through extensive experiments, we demonstrate that our proposed algorithm significantly outperforms competitors.**

I. Introduction

The problem of optimally storing content in a network arises in a broad array of networking applications and systems, including information-centric networks (ICNs) [1], content-delivery networks (CDNs) [2], and wireless/femtocell networks [3], to name a few. It has recently been the focus of several studies that aim to design *cache networks* with optimality guarantees [1]–[7]. Such works optimize either caching decisions alone [2], [4], [5] or caching and routing jointly [2], [6]. Objectives include, e.g., minimizing aggregate transfer costs [4], [5] or queuing delays [7]–[9], maximizing a fairness objective [1], [10] or throughput [11], [12], etc.

Following Ioannidis and Yeh [6], we consider a network in which requests for content generated by customer-facing gateways are routed towards fixed servers, but can be served by intermediate, cache-enabled nodes. The network designer's goal is to determine (a) how to route requests, as well as (b) where to place contents, to minimize overall transfer costs. Even though this problem is NP-hard, Ioannidis and Yeh [6] provide a polytime approximation algorithm, and show that joint optimization of caching and routing decisions can reduce transfer costs by three orders of magnitude, in practice.

This analysis assumes infinite link capacities, which is implausible for real-life networks. We depart by introducing *link capacity constraints*: we assume every edge in the network can carry at most a constant amount of traffic per second. This is clearly more realistic, but also leads to optimization problems of a vastly different nature than [6]. For example, unbounded capacities result in deterministic optimal solutions, whereby each demand is routed over a single, unique path. In contrast, introducing link capacity constraints gives rise to *multi-path* optimal solutions: optimal traffic may split across multiple routes. From a technical standpoint, introducing link capacities drastically changes our optimization problem. In contrast to the vast majority of prior research in the area [3], [6], [8], our constraints no longer form a matroid; this requires a very different algorithm than the one employed by [6].

Several works [13]–[15] consider congestion control in bipartite (one-hop) cache networks, but their network model and guarantees do not apply to arbitrary (multi-hop) topologies. Closer to us, Liu et al. [11] and Kamran et al. [12] provide approximation guarantees under link capacity constraints in arbitrary topologies, but differ in both their objective (throughput maximization) and constraints. In particular, there is no notion of routing costs, as incorporated in our setting. Moreover, none of above works gives rise to the DR-submodular structures we observe in our optimization problem. Overall, these existing algorithms cannot be applied to our setting. Our contributions are as follows:

- We model the problem of joint optimization of caching and routing decisions with link capacity constraints over an arbitrary topology. Our model yields a continuous DR-submodular maximization problem under a set of continuous DR-supermodular constraints.
- The objective is not concave and constraints are not convex. We propose a polynomial-time Lagrangian primal-dual algorithm for this problem. Though the combined, end-to-end algorithm is a heuristic, we show that a $1-\frac{1}{e}$ approximation guarantee holds during primal steps.
- Finally, we conduct extensive experiments over both synthetic and trace-driven networks: our proposed algorithm outperforms several baselines significantly w.r.t. both cache gain and feasibility.

The remainder of this paper is organized as follows. Sec. II introduces the model of cache networks and formulates joint optimization problem. Sec. III describes our analysis of the problem and proposed algorithm. We present numerical experiments in Sec. IV and conclude in Sec. V. The extended version of this paper is available in [16].

II. Model

Network Model and Content Requests. We consider a network represented as a directed, symmetric[1] graph $G(V, E)$.

[1]A directed graph is symmetric when $(i, j) \in E$ implies that $(j, i) \in E$.

The authors gratefully acknowledge support from National Science Foundation grants NeTS-1718355 and CCF-1750539, National Natural Science Foundation of China under Grant 62172054 and the National Key R&D Program of China under Grant 2019YFB1802603.

Content items (e.g., files, or file chunks) of equal size are to be distributed across network nodes. We denote by $\mathcal{C}$ the set of content items, i.e., the *catalog*. The network serves requests for items in $\mathcal{C}$ routed over the G. A request (i, s) is determined by (a) the item $i \in \mathcal{C}$ requested, and (b) the request source $s \in V$. We denote by $\mathcal{R} \subseteq \mathcal{C} \times V$ the set of all requests. For each $i \in \mathcal{C}$, there exists a fixed set of *designated server* nodes $\mathcal{S}_i \subseteq V$, that always store i. A node $v \in \mathcal{S}_i$ permanently stores i in *excess memory outside its cache*. A request (i, s) is routed over a path in G towards a designated server. However, forwarding terminates upon reaching *any intermediate cache* that stores i. At that point, a response carrying i is sent over the reverse path, back to s. Both caching *and* routing decisions are network design parameters, while request arrivals are problem inputs. We define all three below.

Request Arrival Process. Requests arrive according to an i.i.d. process: time is slotted and, at each timeslot $t \in \mathbb{N}$, a random subset $\mathcal{R}(t) \subseteq \mathcal{R}$ of requests occur. We denote by

$$\lambda_{(i,s)} = \mathbf{P}[(i, s) \in \mathcal{R}(t)] \in [0, 1], \quad (i, s) \in \mathcal{R}, \tag{1}$$

the marginal probability that request (i, s) occurs.

Caching Strategies. Each node has a cache that can store a finite number of items. We denote by $c_v \in \mathbb{N}$ the capacity at node $v \in V$. For each node $v \in V$, vector $\boldsymbol{x}_v \in \{0, 1\}^{|\mathcal{C}|}$ indicates v's caching state: $x_{v,i} \in \{0, 1\}$, for $i \in \mathcal{C}$, is the binary variable indicating whether v stores content item i. We assume that vectors $\boldsymbol{x}_v$ are random and independent across $v \in V$. As v can store no more than c_v items, we have:

$$\sum_{i \in \mathcal{C}} x_{v,i} \leq c_v, \text{ for all } v \in V. \tag{2}$$

We define the system's *caching strategy* to be a stationary probability distribution over valid caching states $\boldsymbol{x} \in \{0, 1\}^{|V||\mathcal{C}|}$, i.e., ones that (a) satisfy Eq. (2) and (b) have a product form over $v \in V$. We denote by

$$\xi_{v,i} \equiv \mathbf{P}[x_{v,i} = 1] = \mathbb{E}[x_{v,i}] \in [0, 1], \text{ for } i \in \mathcal{C}, \tag{3}$$

the marginal probability that node v caches item i, and by $\boldsymbol{\xi} = [\xi_{v,i}]_{v \in V, i \in \mathcal{C}} \in [0, 1]^{|V||\mathcal{C}|}$, the corresponding expectation of the caching strategy. By Eq. (2) and Eq. (3):

$$\sum_{i \in \mathcal{C}} \xi_{v,i} \leq c_v, \text{for all } v \in V. \tag{4}$$

Source Routing Strategies. For every request $(i, s) \in \mathcal{R}$, we assume that there exists a set $\mathcal{P}_{(i,s)}$ of *paths* that the request can follow towards a designated server in $\mathcal{S}_i$. A source node s can forward a request among any of these paths; however, responses are constrained to reversely follow the same path as the request they serve. A path p of length $|p| = K$ is a sequence $\{p_1, p_2, \ldots, p_K\}$ of nodes $p_k \in V$. Given a path p and a $v \in p$, let $k_p(v)$ be the position of v in p. Given sets $\mathcal{P}_{(i,s)}$, $(i, s) \in \mathcal{R}$, the *routing state* of a source $s \in V$ w.r.t. request $(i, s) \in \mathcal{R}$ is a vector $\boldsymbol{r}_{(i,s)} \in \{0, 1\}^{|\mathcal{P}_{(i,s)}|}$, where $r_{(i,s),p} \in \{0, 1\}$ is a binary variable indicating whether s selects path $p \in \mathcal{P}_{(i,s)}$. These satisfy:

$$\sum_{p \in \mathcal{P}_{(i,s)}} r_{(i,s),p} = 1, \text{ for all } (i, s) \in \mathcal{R}, \tag{5}$$

indicating that exactly one path is selected. We again assume that $\boldsymbol{r}_{(i,s)}$ are independent random variables across $(i, s) \in \mathcal{R}$.

The system's *routing strategy* to be a stationary distribution over valid routing states, i.e., states that (a) satisfy Eq. (5) and (b) have a product form over $(i, s) \in \mathcal{R}$. For $p \in \mathcal{P}_{(i,s)}$, let

$$\rho_{(i,s),p} \equiv \mathbf{P}[r_{(i,s),p} = 1] = \mathbb{E}[r_{(i,s),p}] \in [0, 1], \tag{6}$$

be the marginal probability that path p is selected by s. Let $P_{\texttt{TOT}} = \sum_{(i,s) \in \mathcal{R}} |\mathcal{P}_{(i,s)}|$, be the total number of paths. Then, the routing strategy is determined by $\boldsymbol{\rho} = [\rho_{(i,s),p}]_{(i,s) \in \mathcal{R}, p \in \mathcal{P}_{(i,s)}} \in [0, 1]^{P_{\texttt{TOT}}}$, where, by Eqs. (5) and (6),

$$\sum_{p \in \mathcal{P}_{(i,s)}} \rho_{(i,s),p} = 1, \text{ for all } (i, s) \in \mathcal{R}. \tag{7}$$

Link Capacities. Every edge $(u, v) \in E$ is associated with a capacity $\mu_{u,v} \geq 0$, indicating the maximum traffic it can sustain: in expectation, the traffic at (u, v) must not exceed $\mu_{u,v}$. Formally, since cache states across nodes in the path $p \in \mathcal{P}_{(i,s)}$ are independent, we have that for all $(u, v) \in E$:

$$\sum_{(i,s) \in \mathcal{R}} \lambda_{(i,s)} \sum_{p \in \mathcal{P}_{(i,s)} : (v,u) \in p} \rho_{(i,s),p} \prod_{k'=1}^{k_p(v)} (1 - \xi_{p_{k'},i}) \leq \mu_{u,v}. \tag{8}$$

Costs and Objective. To capture costs (e.g., latency, money, etc.), we associate a *weight* $w_{u,v} \geq 0$ with each edge $(u, v) \in E$, representing the cost of transferring an item across (u, v). Again, by independence, the expected transfer cost for serving a request $(i, s) \in \mathcal{R}$ given pair $(\boldsymbol{\xi}, \boldsymbol{\rho})$ is:

$$C_{(i,s)}(\boldsymbol{\xi}, \boldsymbol{\rho}) = \sum_{p \in \mathcal{P}_{(i,s)}} \rho_{(i,s),p} \sum_{k=1}^{|p|-1} w_{p_{k+1},p_k} \prod_{k'=1}^{k} (1 - \xi_{p_{k'},i}). \tag{9}$$

Intuitively, Eq. (9) states that $C_{(i,s)}$ includes the cost of an edge (p_{k+1}, p_k) in the path p if (a) p is selected by the routing strategy, and (b) *no* cache preceding this edge in p stores i.

We wish to minimize the total expected transfer cost:

$$\text{Minimize:} \quad C(\boldsymbol{\xi}, \boldsymbol{\rho}) = \sum_{(i,s) \in \mathcal{R}} \lambda_{(i,s)} C_{(i,s)}(\boldsymbol{\xi}, \boldsymbol{\rho}) \tag{10a}$$

$$\text{subj. to:} \quad \text{Eqs. (3), (4), (6), (7), and (8).} \tag{10b}$$

This problem is *NP-hard* [3], [6]. We note that, the constraint set is not a convex polytope, due to Eq. (8), and the objective is not convex. Compared to the setting considered by Ioannidis and Yeh [6], we account for additional capacity constraints via Eq. (8), which in turn lead to the non-convexity of the constraint set.

III. Main Results

Despite the lack of convexity of Problem (10), we show that after an appropriate change of variables the objective can be written as a continuous DR-submodular function [17]. This

gives rise to a primal-dual heuristic, in which primal steps are approximable via a polytime algorithm.

A. Conversion to a Continuous DR-submodular Problem

To convert Problem (10) to a problem amenable through a solution via algorithms that exploit DR-submodularity, we first introduce the auxiliary variables, for all $p \in \mathcal{P}_{(i,s)}, (i,s) \in \mathcal{R}$:

$$\tilde{\rho}_{(i,s),p} = 1 - \rho_{(i,s),p} \in [0,1]. \tag{11}$$

I.e., these are the "complements" of the routing variables; we also denote the corresponding vector comprising these complement variables by $\tilde{\boldsymbol{\rho}} \in [0,1]^{P_{\text{TOT}}}$. Let $C_0 \equiv \sum_{(i,s)\in\mathcal{R}} \lambda_{(i,s)} \sum_{p\in\mathcal{P}_{(i,s)}} \sum_{k=1}^{|p|-1} w_{p_{k+1},p_k}$. Observe that this is a universal constant, not depending or $\boldsymbol{\rho}$ or $\boldsymbol{\xi}$. We define the objective:

$$\begin{aligned} F(\boldsymbol{\xi}, \tilde{\boldsymbol{\rho}}) &= C_0 - C(\boldsymbol{\xi}, 1-\tilde{\boldsymbol{\rho}}) \\ &= \sum_{(i,s)\in\mathcal{R}} \lambda_{(i,s)} \sum_{p\in\mathcal{P}_{(i,s)}} \sum_{k=1}^{|p|-1} w_{p_{k+1},p_k} \cdot (1- \\ &\qquad (1-\tilde{\rho}_{(i,s),p}) \prod_{k'=1}^{k} (1-\xi_{p_{k'}i})), \end{aligned} \tag{12}$$

as the expected *cache gain*. Observe that F is monotone increasing w.r.t. all of its variables. Thus, Prob. (10) is equivalent to the following cache gain maximization problem:

$$\text{Maximize: } F(\boldsymbol{\xi}, \tilde{\boldsymbol{\rho}}) \tag{13a}$$

$$\text{subj. to: } (3), (4), (11) \tag{13b}$$

$$\sum_{p\in\mathcal{P}_{(i,s)}} (1-\tilde{\rho}_{(i,s),p}) = 1, \text{ for all } (i,s) \in \mathcal{R}, \tag{13c}$$

$$G_{u,v}(\boldsymbol{\xi}, \tilde{\boldsymbol{\rho}}) \leq 0, \quad \text{for all } (u,v) \in E, \tag{13d}$$

where we define the *flow* over edge $(u,v) \in E$ to be

$$\lambda_{(u,v)}(\boldsymbol{\xi}, \tilde{\boldsymbol{\rho}}) = \sum_{(i,s)\in\mathcal{R}} \sum_{\substack{p\in\mathcal{P}_{(i,s)}: \\ (v,u)\in p}} \lambda_{(i,s)} (1-\tilde{\rho}_{(i,s),p}) \prod_{k'=1}^{k_p(v)} (1-\xi_{p_{k'},i}), \tag{14}$$

and the *overflow* at $(u,v) \in E$ to be

$$G_{u,v}(\boldsymbol{\xi}, \tilde{\boldsymbol{\rho}}) = \lambda_{(u,v)}(\boldsymbol{\xi}, \tilde{\boldsymbol{\rho}}) - \mu_{u,v}. \tag{15}$$

The objective is not concave, and the constraints involving overflow functions above are not convex. Nevertheless, the following can be shown using the earlier analysis of [6], [17]:

Lemma 1. *Function F, defined in Eq.* (12)*, is non-decreasing and continuous diminishing-returns (DR) submodular, and functions $G_{u,v}$, for all $(u,v) \in E$, defined in Eq.* (15)*, are non-increasing and continuous DR-supermodular.*

Existing algorithms for DR-submodular maximization [17]–[19] do not directly apply to our optimization problem, as they require constraints either being convex or containing at most one supermodular constraint.

Algorithm 1: Primal-Dual Algorithm

Input: $L(\boldsymbol{y}, \boldsymbol{\psi})$, $\mathcal{D}$.
$t \leftarrow 0$, $\boldsymbol{\psi}(0) \leftarrow \mathbf{0}$.
while $t < \tau$ *convergence condition is not met* **do**
  $\boldsymbol{y}(t+1) = \alpha_t \arg\max_{\boldsymbol{y}\in\mathcal{D}} L(\boldsymbol{y}, \boldsymbol{\psi}(t)) + (1-\alpha_t)\boldsymbol{y}(t)$
  $\psi_e(t+1) = [\psi_e(t) + \beta_t G_e(\boldsymbol{y}(t+1))]^+$, for all $e \in E$
  $t \leftarrow t+1$
end
return $\boldsymbol{y}_k$

B. Lagrangian and Duality

Consider the Lagrangian:

$$L(\boldsymbol{\xi}, \tilde{\boldsymbol{\rho}}, \boldsymbol{\psi}) = F(\boldsymbol{\xi}, \tilde{\boldsymbol{\rho}}) - \textstyle\sum_{e\in E} \psi_{u,v} \cdot G_{u,v}(\boldsymbol{\xi}, \tilde{\boldsymbol{\rho}}), \tag{16}$$

where vector $\boldsymbol{\psi} = [\psi_{u,v}]_{(u,v)\in E}$ is the non-negative *dual variables* associated with the constraint (13d). Intuitively, the Lagrangian function L penalizes the infeasibility of the link capacity constraints. The following theorem is an immediate consequence of Lemma 1:

Theorem 1. *Function L is non-decreasing and continuous DR-submodular.*

To motivate our approach, assume we were given proper dual variables $\boldsymbol{\psi}$. Then, optimizing the Lagrangian converts the cache gain maximization problem (13) to the following:

$$\text{Maximize: } L(\boldsymbol{\xi}, \tilde{\boldsymbol{\rho}}, \boldsymbol{\psi}) \tag{17a}$$

$$\text{subj. to: } \boldsymbol{\xi}, \tilde{\boldsymbol{\rho}} \in \mathcal{D} \tag{17b}$$

where $\mathcal{D}$ is the set defined by constraints: (3), (4), (11), and (13c). Prob. (17) has a non-decreasing, continuous DR submodular objective, and convex constraints $\mathcal{D}$.

C. Primal-Dual Algorithm

Motivated by the above observation, we propose solving Prob. (13) via a primal-dual algorithm. The primal steps of the algorithm reduce to solving, Prob. (17) which is a monontone DR-submodular optimization problem with affine constraints; though not down-closed convex or even not convex, we are able to solve this via a polytime algorithm within a $1-1/e$ approximation guarantee.

1) Algorithm Overview: For brevity, we join $\boldsymbol{\xi}(t)$ and $\tilde{\boldsymbol{\rho}}(t)$ as one variable $\boldsymbol{y}(t) = (\boldsymbol{\xi}(t), \tilde{\boldsymbol{\rho}}(t))$, and denote by it *primal variables*. The primal-dual algorithm starts from $\boldsymbol{\psi}(0) = \mathbf{0}$ and iterates over:

$$\boldsymbol{y}(t+1) = \alpha_t \arg\max_{\boldsymbol{y}\in\mathcal{D}} L(\boldsymbol{y}, \boldsymbol{\psi}(t)) + (1-\alpha_t)\boldsymbol{y}(t), \tag{18a}$$

$$\psi_e(t+1) = [\psi_e(t) + \beta_t G_e(\boldsymbol{y}(t+1)]^+, \text{for all } e \in E, \tag{18b}$$

where $\alpha_t = \frac{2}{t+2}$ is the parameter of momentum, $\beta_t = \frac{c}{\sqrt{t}}$ is the step size, c is a constant, and $[z]^+ = \max\{z, 0\}$. We summarize this also in Alg. 1, and discuss each step in detail below:

Algorithm 2: Frank-Wolfe variant for $L(\boldsymbol{y}, \boldsymbol{\psi}(t))$

Input: $L(\boldsymbol{y}, \boldsymbol{\psi}(t))$, $\mathcal{D}'$, step size $\gamma \in (0, 1]$.

$\tau \leftarrow 0$, $k \leftarrow 0$, $\boldsymbol{y}_0 \leftarrow \mathbf{0}$.

while $\tau < 1$ **do**

  $\boldsymbol{v}_k \leftarrow \arg\max_{\boldsymbol{v} \in \mathcal{D}'} \langle \boldsymbol{v}, \nabla L(\boldsymbol{y}, \boldsymbol{\psi}(t)) \rangle$

  $\gamma_k \leftarrow \min\{\gamma, 1 - \tau\}$

  $\boldsymbol{y}_{k+1} = \boldsymbol{y}_k + \gamma_k \boldsymbol{v}_k$, $\tau \leftarrow \tau + \gamma_k$, $k \leftarrow k + 1$

end

return $\boldsymbol{y}_k$

Primal Step (18a): The primal step updates primal variables $\boldsymbol{y}$ given dual variables $\boldsymbol{\psi}$. It first solves Prob. (17); then, it utilizes a momentum parameter to alleviate the change of primal variables. Since $\boldsymbol{y}(t+1)$ is a convex combination of two points in feasible set $\mathcal{D}$, it still lies in $\mathcal{D}$. We describe how to solve (17) approximately in Sec. III-C2. The smoothing process via the momentum is crucial, as it helps with the convergence of the algorithm: we observe this experimentally in Sec. IV-E.

Dual Step (18b): Finally, the dual step updates dual variables $\boldsymbol{\psi}$ given primal variables $\boldsymbol{y}$ via dual ascent.

2) Primal Variables via Frank-Wolfe Algorithm: We solve Problem (18a) through a variant of Frank-Wolfe algorithm, summarized in Alg. 2. Starting from $\boldsymbol{y}_0 = (\boldsymbol{\xi}_0, \tilde{\boldsymbol{\rho}}_0) = \mathbf{0}$, the algorithm iterates over:

$$\boldsymbol{v}_k = \arg\max_{\boldsymbol{v} \in \mathcal{D}'} \langle \boldsymbol{v}, \nabla L(\boldsymbol{y}, \boldsymbol{\psi}(t)) \rangle \tag{19a}$$

$$\boldsymbol{y}_{k+1} = \boldsymbol{y}_k + \gamma_k \boldsymbol{v}_k, \tag{19b}$$

where γ_k is the proper step size satisfying $\sum_k \gamma_k = 1$, gradient

$$\nabla_i L(\boldsymbol{y}, \boldsymbol{\psi}(t)) = L(\boldsymbol{y}, \boldsymbol{\psi}(t)|_{y_i=1}) - L(\boldsymbol{y}, \boldsymbol{\psi}(t)|_{y_i=0}), \tag{20}$$

and $\mathcal{D}'$ is the set:

$$(3), (4), (11), \tag{21a}$$

$$\textstyle\sum_{p \in \mathcal{P}_{(i,s)}} (1 - \tilde{\rho}_{(i,s),p}) \geq 1, \quad \text{for all } (i,s) \in \mathcal{R}, \tag{21b}$$

The difference between $\mathcal{D}$ and $\mathcal{D}'$ lies in having inequalities in Eq. (21b), which relaxes Eq. (13c). Note that $\mathcal{D}'$ is a down-closed convex set while $\mathcal{D}$ is not. The following theorem states the approximation guarantee we attain for this algorithm w.r.t. the (non-relaxed) Prob. (17).

Theorem 2. *Let $\boldsymbol{y}^*$ be an optimal solution to Prob.* (17)*, and $\boldsymbol{y}_{FW}$ be the output of the Frank-Wolfe variant Alg. 2. Then, $\boldsymbol{y}_{FW}$ belongs to $\mathcal{D}$, and given any $\boldsymbol{\psi}$:*

$$L(\boldsymbol{y}_{FW}, \boldsymbol{\psi}) + C \geq (1 - \frac{1}{e})(L(\boldsymbol{y}^*, \boldsymbol{\psi}) + C) - \frac{M}{2K}, \tag{22}$$

where constant $C = \sum_{e \in E} \psi_e (\sum_{(i,s) \in \mathcal{R}} \lambda_{(i,s)} - \mu_e)$, $M = 2L(\mathbf{1}, \boldsymbol{\psi})(|V||\mathcal{C}| + P_{\text{TOT}})^2$ is the Lipschitz continuous constant, and $K = \frac{1}{\gamma}$ is the number of iterations.

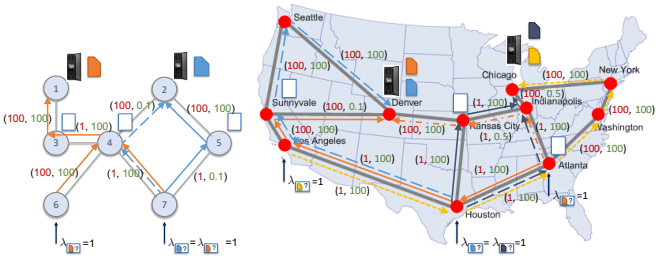
Proof Sketch. Frank-Wolfe variant algorithm shown in Alg. 2 is a classic method [17] for:

$$\max_{\boldsymbol{y} \in \mathcal{D}'} L(\boldsymbol{y}, \boldsymbol{\psi}), \tag{23}$$

which is a continuous DR-submodular maximization problem under down-closed convex constraint. We first prove that constraints $\mathcal{D}'$ are binding, i.e., there exists an optimal point $y^{**} = \arg\max_{\boldsymbol{y} \in \mathcal{D}'} L(\boldsymbol{y}, \boldsymbol{\psi})$, such that the inequality (21b) in $\mathcal{D}'$ holds with equality (13c) in $\mathcal{D}$, hence, $\boldsymbol{y}^{**} \in \mathcal{D}$. This part is proved by contradiction. Thus, we can infer that $L(\boldsymbol{y}^{**}, \boldsymbol{\psi}) = L(\boldsymbol{y}^*, \boldsymbol{\psi})$. Similarly, $\boldsymbol{y}_{\text{FW}} \in \mathcal{D}$ is also in $\mathcal{D}'$. Finally, to provide an optimality factor, we offset L by a constant C. We provide a $1 - \frac{1}{e}$ performance guarantee according to Corollary 1 in [17]. □

Note that, if we choose a large enough K, the offset $\frac{M}{2K}$ can become arbitrary small. The constant C is necessary to obtain an approximation guarantee as, in general, the Lagrangian (16) can become negative, and adding this term ensures positivity.

In practice, we found that setting the scaling factor c in β_t, defined in (18b), so that the Lagrangian remains always positive is preferable experimentally: in some sense, ensuring the positivity of L strikes a good balance between the two components (cache gain and constraint penalization) of the objective. In contrast, a negative Lagrangian indicates a high penalization of infeasibility, and a discount of the cache gain. Furthermore, given a gradient, algorithm (19) requires polynomial time in the number of constraints and variables, which are $O(|V||\mathcal{C}| + |E||\mathcal{R}|)$. We iterate (19) at most $O(|V||\mathcal{C}|)$ times [8].



(a) `Ex1` with $\kappa = 1$ (b) `Abilene1` with $\kappa = 1$

Fig. 1: Topologies and parameters of `Ex1` and `Abilene1` with designed requests and bandwidths. There is a pair (red, green) for edge (u,v), where the first red number is the weight $w_{(v,u)}$ and the second green number is the link capacity $\mu_{(v,u)}$.

IV. Experiments

We conduct both *synthetic* and *trace-driven* experiments.

A. Synthetic Experiment Setup

Networks. To evaluate our proposed algorithm, we perform experiments over five synthetic graphs, and a counter example designed to demonstrate suboptimality of competitors (`Ex`). We also experiment with three backbone network topologies. The parameters of different topologies are summarized in Tab. I. The weights of each edge $w_{u,v}$, $(u,v) \in \mathcal{E}$ are selected uniformly at random (u.a.r.) from 1 to 100. Each node $v \in V$ has c_v storage to cache items from a catalog of size $|\mathcal{C}|$. Each

TABLE I: Graph Topologies and Experiment Parameters

Graph	$\|V\|$	$\|E\|$	$\|\mathcal{Q}\|$	$\|\mathcal{R}\|$	$\|\mathcal{P}_{(i,s)}\|$	$\|c_v\|$	w	$\|\mathcal{C}\|$	F^1_{PD}	F^3_{PD}
synthetic topology experiments										
Erdős-Rényi (ER)	100	1044	10	4949	1-5	10-20	1-100	1000	2314.9	2318.1
balanced tree (BT)	364	726	10	4988	1-5	10-20	1-100	1000	1665.2	1666.3
hypercube (HC)	128	896	10	4960	1-5	10-20	1-100	1000	3228.5	3229.4
grid_2d (grid)	100	360	10	4954	1-5	10-20	1-100	1000	5753.2	5753.7
small-world (SW) [20]	100	503	10	4953	1-5	10-20	1-100	1000	4482.1	4484.3
Ex1	7	14	2	3	1-2	0-1	1-100	2	398.8	388.7
Ex2									351.8	365.4
backbone network experiments [21]										
GEANT	22	66	4	4761	1-5	10-20	1-100	1000	4436.2	4440.7
Deutsche Telekom (DT)	68	546	4	4929	1-5	10-20	1-100	1000	2014.7	2030.0
Abilene1	11	28	3	4	1-2	0-1	1-100	4	814.3	901.0
Abilene2									761.4	789.4
trace-driven experiments										
KS1	152	22952	101	1988	1-5	25-3195	1-100	526	19938.5	19946.7
KS2	152	22952	103	4963	1-5	50-6390	1-100	1207	35353.1	35349.4

item $i \in \mathcal{C}$ is stored permanently in one designated server $\mathcal{S}_i$ which is picked u.a.r. from V; the item is stored outside the designated server's cache. For Ex and Abilene, we select parameters in a way demonstrated in Figs. 1a and 1b, respectively.

Requests. We generate requests synthetically as follows. We select u.a.r. a set of $\mathcal{Q}$ nodes from V as the possible query nodes. The set of requests $\mathcal{R} \subseteq \mathcal{C} \times \mathcal{Q}$ is then generated by sampling from the set $\mathcal{C} \times \mathcal{Q}$, u.a.r. For each such request $(i, s) \in \mathcal{R}$, we select the request arrival probability $\lambda_{(i,s)}$ according to a Zipf distribution with parameter 1.2. For each request $(i, s) \in \mathcal{R}$, we generate at most $|\mathcal{P}_{(i,s)}|$ paths from the source $s \in V$ to the designated server $\mathcal{S}_i$, where the source s and the designated server $\mathcal{S}_i$ are not the same node. In all cases, this path set includes the shortest path to the designated server. We consider only paths with stretch at most 4; that is, the maximum cost of a path in $\mathcal{P}_{(i,s)}$ is at most 4 times the cost of the shortest path to the designated source. We follow a different synthetic request generation process for Ex and Abilene, based on the "hard" examples we describe in [16], on which we prove that competitors may fail to produce feasible solutions.

Link Capacities. To control the level of congestion in the network, we determine link capacities $\mu_{u,v}, (u, v) \in E$ as follows. We first randomly sample c_v items i and set $\xi_{v,i} = 1$, for all $v \in V$, and set $\tilde{\rho}_{(i,s),p} = \frac{1}{|\mathcal{P}_{(i,s)}|}$, for all $p \in \mathcal{P}_{(i,s)}, (i, s) \in \mathcal{R}$. Then, we set the link capacities as $\mu_{u,v} = \kappa\lambda_{(u,v)}(\boldsymbol{\xi}, \tilde{\boldsymbol{\rho}})$ correspondingly, where $\lambda_{(u,v)}$ is the flow on edge (u, v), given by Eq. (14), and $\kappa \geq 1$ is a *looseness coefficient*: the higher κ is, the easier it is to satisfy the link capacity constraints.

B. Trace-Driven Experiment Setup

Finally, we also conduct trace-driven simulations using data from a short video application, Kuaishou (KS) [22]. This comprises more than 8 million requests of 2 million items/videos reaching 488 Kaishou edge servers deployed at 31 provinces in China within 5 mins. The network topology (including nodes, links, and link and cache capacities) are determined from an actual cache deployment by Kuaishou. We preprocess the data to create two instances (KS1 and KS2), whose statistics are summarized in Tab. I, as follows.

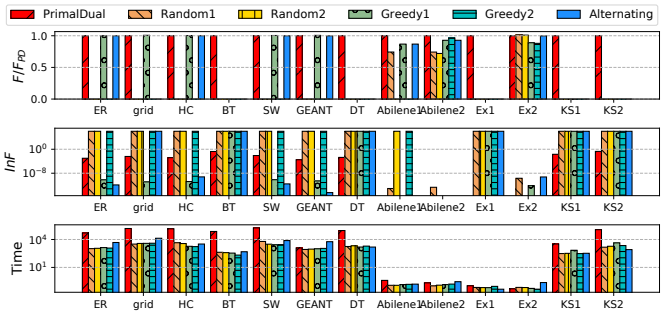
We select the largest connected subgraph, and utilize $\frac{1}{4}$ and $\frac{1}{2}$ of caches equipped by each node for our experiments KS1 and KS2, respectively. In KS1, we restrict traffic of top 2000 popular requests, while in KS2 we restrict traffic to the top 5000 popular requests. We again generate all paths of stretch at most 4; we drop any request that does not contain any paths in the largest connected component, leading to the numbers reported in Table I. We use these to compute request probabilities $\lambda_{(i,s)} \in [0, 1]$; to do so, we normalize each request frequency by the frequency of the most popular request. As we limit traffic to a subset of the entire demand, we scale link capacities in KS1 and KS2 both by $\frac{1}{2250}$.

C. Algorithms

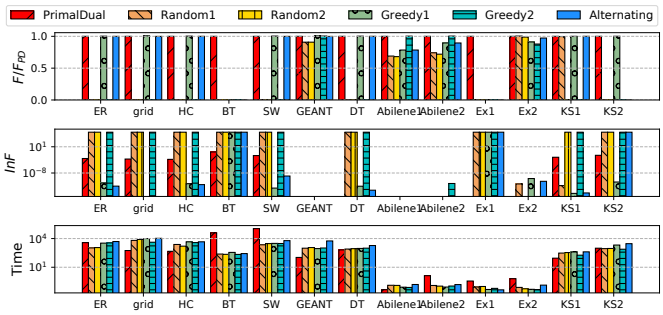
We implement our algorithm and several competitors[2] for comparison purposes. Our main building blocks when constructing competitors are combinations of algorithms that make caching and routing decisions separately.

In particular, building blocks for caching are: (a) *uniform caching*, whereby cache contents are selected uniformly among requests that traverse the cache, (b) *greedy caching*, whereby the greedy algorithm [23] is used to allocate items to caches, and (c) *Frank-Wolfe variant caching*, where the Frank-Wolfe variant algorithm [17] is used to determine cache contents; all three variants (a)–(c) are classic, but *ignore edge capacity constraints*. The classic greedy algorithm is a 1/2 approximation [8], [23], while the Frank-Wolfe variant [17] achieves $1 - 1/e$. We combine these caching algorithms with *optimal routing*, which amounts to fixing a caching strategy, and computing routing decisions by solving Prob. (13) w.r.t. routing decisions alone; this is a convex optimization problem with affine constraints, and can be solved in polynomial time.

[2]Our implementation is publicly available at https://github.com/neu-spiral/CacheRateNetwork.



(a) Looseness coefficient $\kappa = 1$



(b) Looseness coefficient $\kappa = 3$

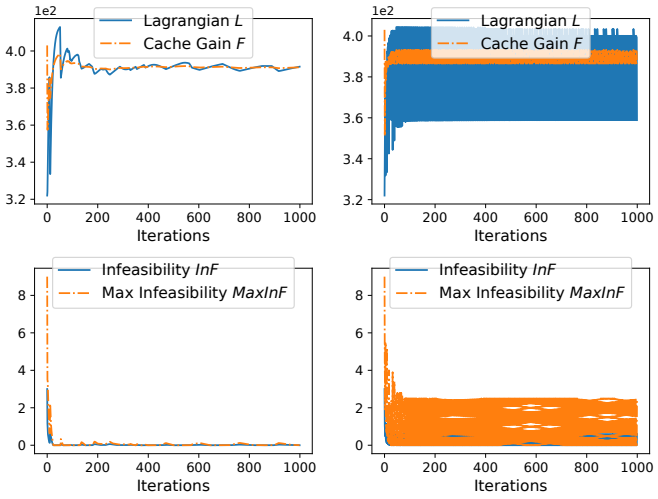
Fig. 2: Comparison w.r.t. gain, `InF` and running time. Missing bars in cache gain plots indicate infeasibility, while missing bars in `InF` plots indicate 0 violation. `PrimalDual` and competitors perform very well when they are feasible. Nevertheless, `PrimalDual` is always feasible for all topologies, while competitors fail to get a feasible solution in some cases. This comes at the cost of the increased complexity of `PrimalDual`, reflected also on the running time. However, when κ is large, i.e., $\kappa = 3$, as `PrimalDual` converge faster, it takes even less execution time compared to competitors.

Overall, we implement the following combinations of these building blocks: `Random1`, that uses uniform caching first and then optimal routing; `Random2`, that performs optimal routing under empty caches first, and then performs uniform caching; `Greedy1`, that uses greedy caching first and then optimal routing; `Greedy2`, that uses optimal routing under empty caches first, and then greedy caching; `Alternating`, that alternates (until convergence) between obtaining a caching state via the Frank-Wolfe variant algorithm and optimal routing; and `PrimalDual` is demonstrated in (18). Additional implementation details are described in [16]. In [16], we prove that *all of the above competitors* (`Random1`–`Alternating`) can lead to arbitrarily suboptimal solutions.

D. Performance Metrics

We use cache gain, defined in Eq. (12), as one metric to measure the performance of different algorithms. Also, we define an *Infeasibility* metric to measure how much solutions violate link capacity constraints. Intuitively, we measure infeasibility as the average overflow, normalized by edge capacities, across all active edges in the network. Formally:

$$\texttt{InF} = \frac{1}{|E'|} \sum_{(u,v)\in E'} \frac{G_{u,v}(\boldsymbol{\xi}, \tilde{\boldsymbol{\rho}}) \mathbb{1}_{G_{u,v}(\boldsymbol{\xi},\tilde{\boldsymbol{\rho}})>0}}{\mu_{u,v}}, \tag{24}$$



(a) with momentum and $\kappa = 1$ (b) without momentum and $\kappa = 1$

Fig. 3: Convergence over topology `Ex1`, w.r.t. the cache gain, the Lagrangian L, and infeasibility metrics `InF` and `MaxInf`. With momentum, algorithm converges faster and smoother.

where overflow $G_{u,v}(\boldsymbol{\xi}, \tilde{\boldsymbol{\rho}})$ is defined in Eq. (15), and $E' = \{e \in E : \lambda_e((\boldsymbol{\xi}, \tilde{\boldsymbol{\rho}}) > 0\}$ is the set of edges with non-zero flow, and flow λ_e is defined in Eq. (14). We say algorithms `Alternating` and `PD` algorithm converge, when `InF` ≤ 0.001 and cache gain changes less than 0.001 compared to the last iteration. We also report `MaxInF`, which is the maximum rather than average over E'. Clearly, larger `InF`/`MaxInf` indicates more violations and worse performance. For our algorithm `PrimalDual`, we expect some negligible edge capacity constraint violation, of the order of `InF` $\sim 10^{-2}$. Whenever `CVXOPT` fails to find a feasible solution, we are unable to compute this score, so we set `InF` $= 10^6$, to indicate a severe feasibility failure.

E. Experiment Results

Different Topologies. We first compare the proposed algorithm (`PrimalDual`) with baselines in terms of the normalized cache gain $\frac{F}{F_{\texttt{PD}}}$, Infeasibility `InF`, and running time of algorithms, shown in Fig. 2. The cache gain $F_{\texttt{PD}}$ is obtained by `PrimalDual` algorithm, and its value is reported in Tab. I: $F^1_{\texttt{PD}}$ is the cache gain when $\kappa = 1$, while $F^3_{\texttt{PD}}$ when $\kappa = 3$. When algorithms obtain no feasible solutions, normalized cache gain and infeasibility are set 0 and 10^6, correspondingly. Observe that `PrimalDual`, `Greedy1`, `Greedy2` and `Alternating` behave great w.r.t. cache gain when they are feasible. However, even though `PrimalDual` *always produces a feasible solution* (with `InF` $\sim 10^{-2}$ consistently), solutions of other algorithms are infeasible in some topologies. This is because `PrimalDual` jointly optimizes both caching and routing decisions. In other words, in every intermediate step, it takes link capacity constraints into consideration. In contrast, competitors decouple routing and caching optimization, ignoring link capacity constraints in the latter. This verifies the suboptimality of competitors. These advantages of

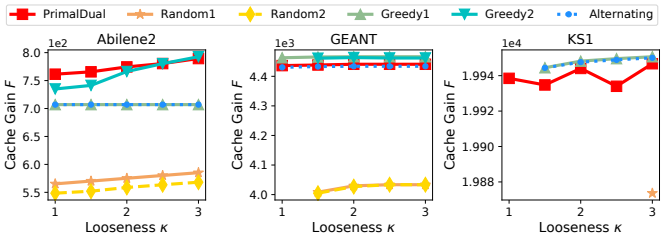


Fig. 4: Effect of looseness over `Abilene2`, `GEANT` and `KS1`, respectively. `PrimalDual` is best or competitive with other algorithms, but also finds a feasible solution for a wider range of κ values.

`PrimalDual` come at the cost of increased running time; nevertheless, with larger looseness κ, `PrimalDual` converges faster, sometimes even outperforming simpler methods.

Convergence. We focus on `Ex1` to understand the convergence of proposed `PrimalDual`. Fig. 3a demonstrates the convergence with momentum (defined in Eq. (18a)). Both cache gain and infeasibility converge smoothly and quickly. On the other hand, without momentum, i.e., for $\alpha_t = 1$, both cache gain and infeasibility exhibit jitter, as shown in Fig. 3b. Algorithms without momentum tend to converge to a more infeasible solution. Overall, incorporating momentum in primal steps avoids oscillations in primal variables and promotes faster and smoother convergence.

Effect of Looseness. Figs. 2, 4 and Tab. I all present results with different looseness coefficient κ. When looseness κ is small, i.e., link capacity constraints are strict and hard to satisfy, competitors are more likely to lead to infeasibility. It is clear from Fig. 2 and Tab. I that, in general, higher κ leads to higher cache gain, less infeasibility and faster convergence/less execution time. This is also indicated directly in Fig. 4: if algorithms have no results at some κ, this indicates infeasibility. In contrast, although not always obtaining the highest cache gain, our proposed `PrimalDual` *always yields a solution*, and is near-optimal.

V. Conclusion

We jointly optimize both caching and routing decisions under bounded link capacity constraints over an arbitrary network. We propose a poly-time primal-dual algorithm, where only primal steps have an approximation guarantee. We use a momentum method to alleviate sharp changes in primal variables. Instead, we could explore a proximal method [24], [25] to realize it. As we only provide approximation guarantees for primal steps, another direct and crucial future direction is to propose an algorithm with end-to-end optimality guarantees.

References

[1] L. Wang, G. Tyson, J. Kangasharju, and J. Crowcroft, "Faircache: Introducing fairness to ICN caching," in *2016 IEEE 24th International Conference on Network Protocols (ICNP)*, 2016, pp. 1–10.

[2] M. Dehghan, A. Seetharamz, T. He, T. Salonidis, J. Kurose, and D. Towsley, "Optimal caching and routing in hybrid networks," in *2014 IEEE Military Communications Conference*. IEEE, 2014, pp. 1072–1078.

[3] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, "Femtocaching: Wireless content delivery through distributed caching helpers," *Transactions on Information Theory*, vol. 59, no. 12, pp. 8402–8413, 2013.

[4] Y. Wang, Z. Li, G. Tyson, S. Uhlig, and G. Xie, "Optimal cache allocation for content-centric networking," in *2013 21st IEEE International Conference on Network Protocols (ICNP)*. IEEE, 2013, pp. 1–10.

[5] S. Ioannidis and E. Yeh, "Adaptive caching networks with optimality guarantees," in *ACM SIGMETRICS*, 2016.

[6] ——, "Jointly optimal routing and caching for arbitrary network topologies," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 6, pp. 1258–1275, 2018.

[7] Y. Li and S. Ioannidis, "Universally stable cache networks," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 546–555.

[8] M. Mahdian, A. Moharrer, S. Ioannidis, and E. Yeh, "Kelly cache networks," *IEEE/ACM Transactions on Networking*, vol. 28, no. 3, pp. 1130–1143, 2020.

[9] Y. Li and S. Ioannidis, "Cache networks of counting queues," *IEEE/ACM Transactions on Networking*, 2021.

[10] Y. Liu, Y. Li, Q. Ma, S. Ioannidis, and E. Yeh, "Fair caching networks," *ACM SIGMETRICS Performance Evaluation Review*, vol. 48, no. 3, pp. 89–90, 2021.

[11] B. Liu, K. Poularakis, L. Tassiulas, and T. Jiang, "Joint caching and routing in congestible networks of arbitrary topology," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 10 105–10 118, 2019.

[12] K. Kamran, A. Moharrer, S. Ioannidis, and E. Yeh, "Rate allocation and content placement in cache networks," in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*. IEEE, 2021, pp. 1–10.

[13] M. Dehghan, A. Seetharam, B. Jiang, T. He, T. Salonidis, J. Kurose, D. Towsley, and R. Sitaraman, "On the complexity of optimal routing and content caching in heterogeneous networks," in *IEEE INFOCOM 2015-IEEE Conference on Computer Communications*. IEEE, 2015, pp. 936–944.

[14] K. Poularakis, J. Llorca, A. M. Tulino, I. Taylor, and L. Tassiulas, "Service placement and request routing in mec networks with storage, computation, and communication constraints," *IEEE/ACM Transactions on Networking*, 2020.

[15] K. Poularakis, G. Iosifidis, and L. Tassiulas, "Approximation caching and routing algorithms for massive mobile data delivery," in *GLOBECOM*, 2013.

[16] Y. Li, Y. Zhang, S. Ioannidis, and J. Crowcroft, "Jointly optimal routing and caching with bounded link capacities," 2023. [Online]. Available: https://arxiv.org/abs/2302.02508

[17] A. A. Bian, B. Mirzasoleiman, J. Buhmann, and A. Krause, "Guaranteed non-convex optimization: Submodular maximization over continuous domains," in *Artificial Intelligence and Statistics*. PMLR, 2017, pp. 111–120.

[18] H. Hassani, M. Soltanolkotabi, and A. Karbasi, "Gradient methods for submodular maximization," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 5843–5853.

[19] R. K. Iyer and J. A. Bilmes, "Submodular optimization with submodular cover and submodular knapsack constraints," *Advances in neural information processing systems*, vol. 26, 2013.

[20] J. Kleinberg, "The small-world phenomenon: An algorithmic perspective," in *STOC*, 2000.

[21] D. Rossi and G. Rossini, "Caching performance of content centric networks under multi-path routing (and more)," Telecom ParisTech, Tech. Rep., 2011.

[22] "Kuaishou," 2022. [Online]. Available: https://www.kuaishou.com

[23] A. Krause and D. Golovin, "Submodular function maximization." 2014.

[24] J. Bolte, S. Sabach, and M. Teboulle, "Proximal alternating linearized minimization for nonconvex and nonsmooth problems," *Mathematical Programming*, vol. 146, no. 1, pp. 459–494, 2014.

[25] S. Bitterlich, R. I. Boţ, E. R. Csetnek, and G. Wanka, "The proximal alternating minimization algorithm for two-block separable convex optimization problems with linear constraints," *Journal of Optimization Theory and Applications*, vol. 182, no. 1, pp. 110–132, 2019.